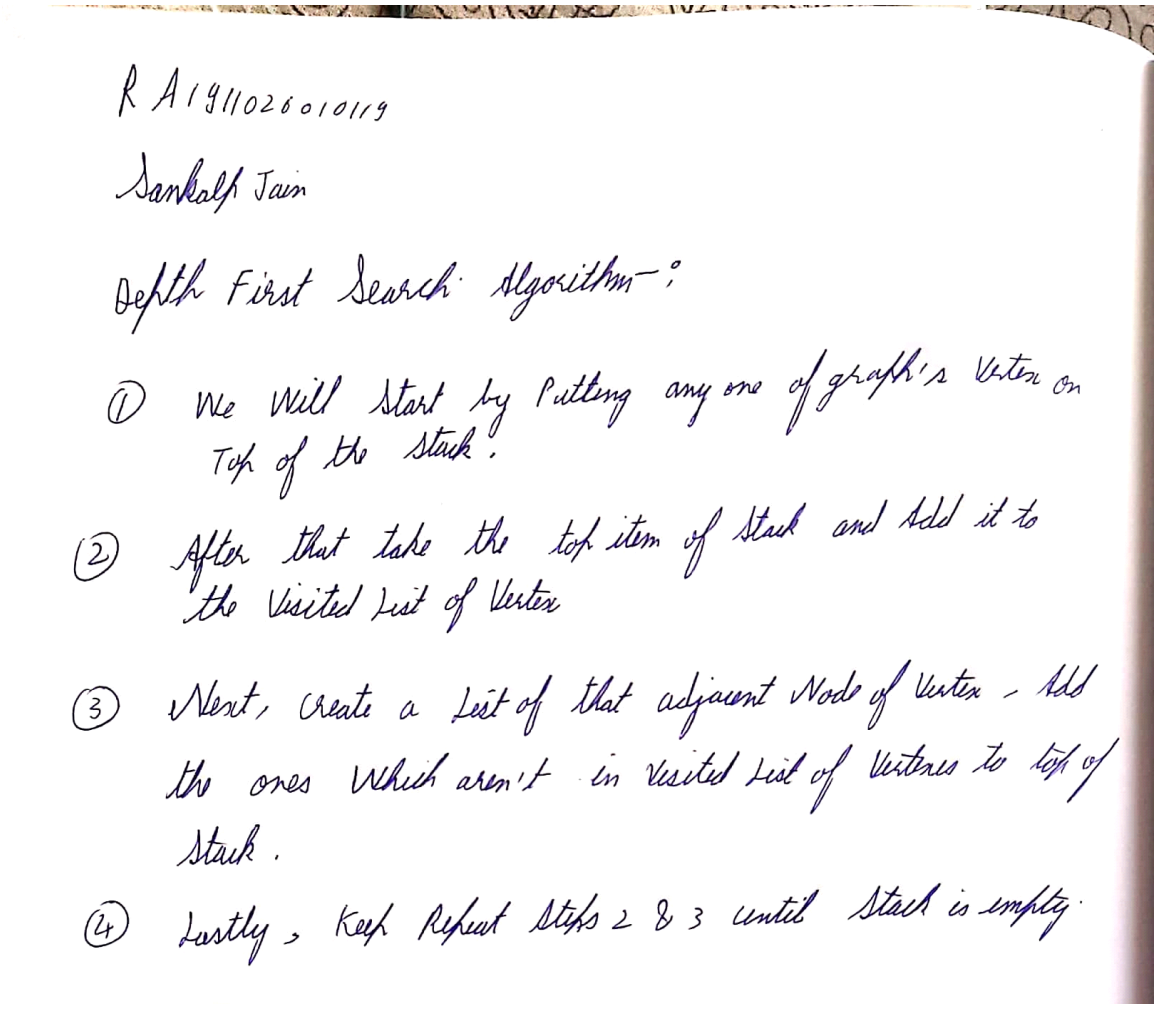
Exp-4
Depth and breadth first search

name-sankalp jain
Rego no-RA1911026010119

Depth first search and Breath first search

AIM- To find the shortest path using BFS(Breadth First Search) uses Queue data structure and depth first search using stack data structure

ALGORITHM-


RA1911026010119

Sankalp Jain

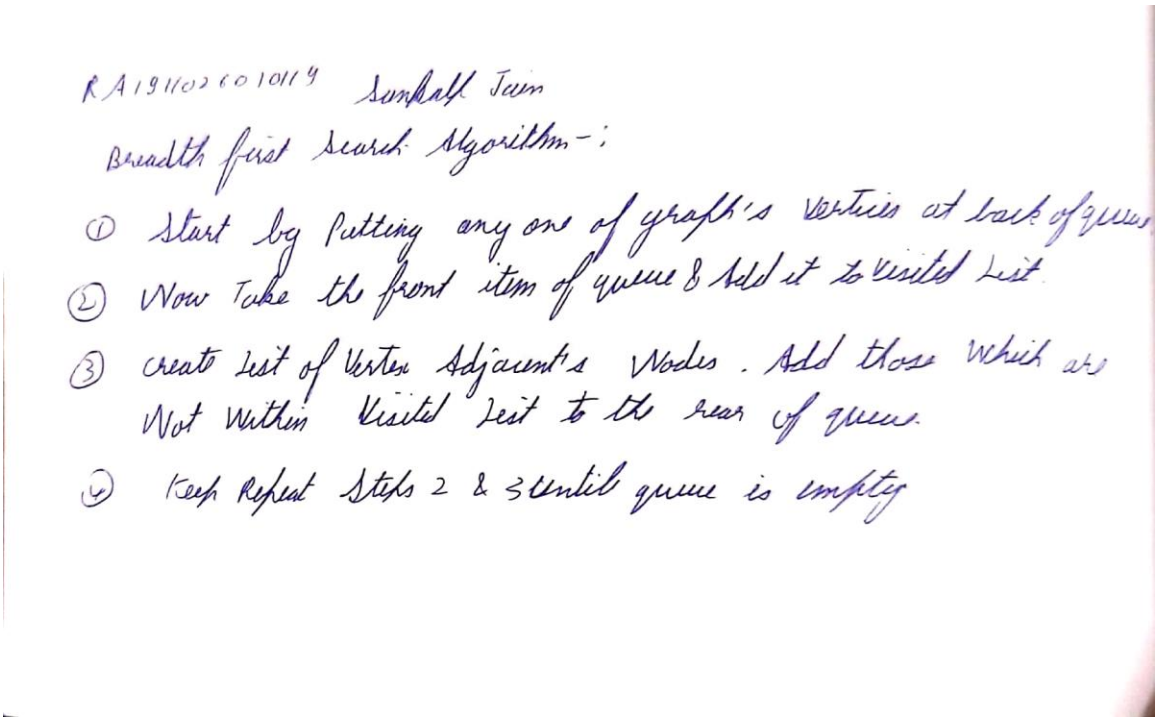
Depth First Search Algorithm :-

1. We will start by putting any one of graph's vertex on top of the stack.
2. After that take the top item of stack and add it to the visited list of vertex
3. Next, create a list of that adjacent Node of vertex - add the ones which aren't in visited list of vertices to top of stack.
4. Lastly, Keep Repeat steps 2 & 3 until stack is empty.

RA1911027010119 Sonkalp Jain

Breadth first search Algorithm -:

1. Start by putting any one of graph's vertices at back of queue.
2. Now Take the front item of queue & add it to visited list.
3. create list of vertex adjacent's nodes. Add those which are not within visited list to the rear of queue.
4. Keep Repeat steps 2 & 3 until queue is empty

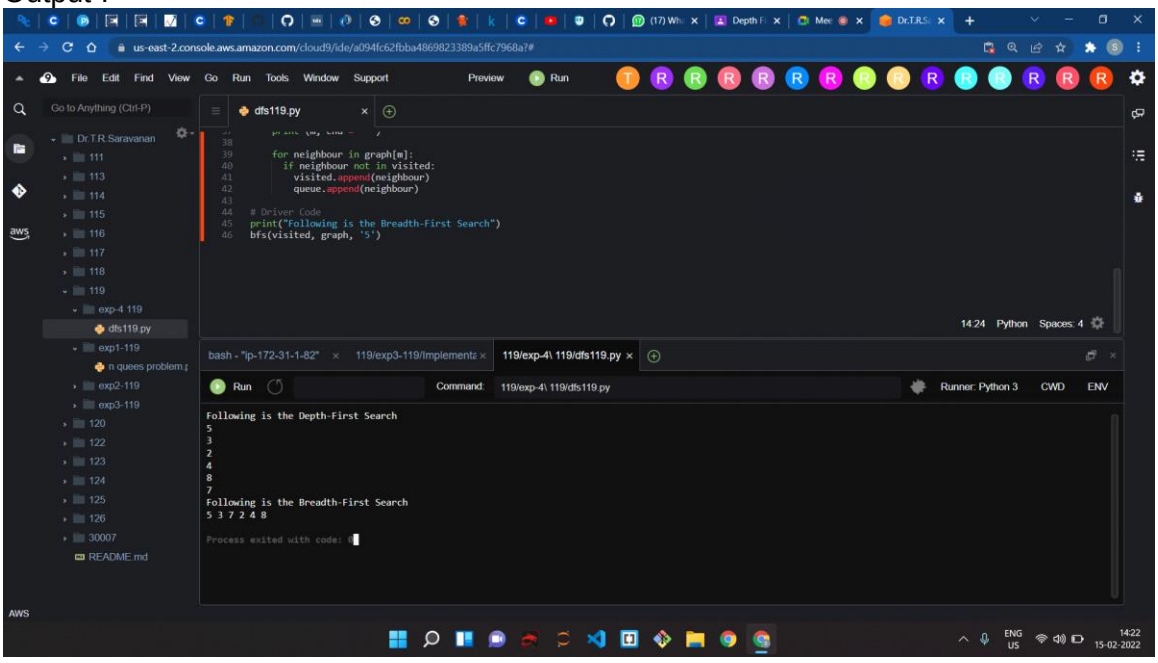


Code-:

```
graph = {
  '5' : set(['3','7']),
  '3' : set(['2', '4']),
  '7' : set(['8']),
  '2' : set([]),
  '4' : set(['8']),
  '8' : set([])
}

def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)

    print(start)

    for next in graph[start] - visited:
        dfs(graph, next, visited)
    return visited



print("Following is the Depth-First Search")
dfs(graph, '5')
```

```
visited = [] # List for visited nodes.
queue = []        #Initialize a queue

def bfs(visited, graph, node): #function for BFS
  visited.append(node)
  queue.append(node)

  while queue:                # Creating loop to visit each node
    m = queue.pop(0)
    print (m, end = " ")

    for neighbour in graph[m]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

# Driver Code
print("Following is the Breadth-First Search")
bfs(visited, graph, '5')
```

Output-:



RESULT-Hence we sucessfully found the shortest path using dfs and bfs